1) Кластерный анализ проводится с помощью модифицированной версии свободно распространяемого программного пакета GROMACS.
   Для установки модифицированной версии программы GROMACS версии 5.1.4 в каталог пользователя на системе Linux необходимо выполнить последовательность команд, приведенную ниже.
   wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-5.1.4.tar.gz
   tar xfz gromacs-5.1.4.tar.gz
   wget http://media.biophys.msu.ru/gmx_cluster-5.1.4.patch
   patch -p0 < gmx_cluster-5.1.4.patch
   cd gromacs-5.1.4 && mkdir build && mkdir run && cd build
   cmake .. -DGMX_FFT_LIBRARY=fftpack -DCMAKE_INSTALL_PREFIX=`pwd`/../run
   make install
   source ../run/bin/GMXRC
   Перед осуществлением кластерного анализа желательно рассчитать матрицу среднеквадратичных расстояний между всеми структурами в анализируемом ансамбле структур.
   gmx rms -s topol.pdb -f frames.xtc -fit none -o rms.xvg -m matrix.xpm -bin matrix.dat
   Для осуществления процедуры кластерного анализа нужно выполнить команду:
   gmx cluster -method density -M 100 -dmb matrix.dat -s topol.pdb -f frames.xtc -nofit -n topol.ndx -g clusters.log -rd clusters.xvg -ord ordered.xtc -cl clusters.pdb
   Параметр M выбирается порядка 1-5% от общего количества анализируемых структур.

2) Для получения файла, содержащего координаты атомов всех структур определенного кластера потребуется создание индекс-файла, содержащего номера всех кадров этого кластера из “ordered.xtc”. Для получения такого файла вы можете использовать приведенный ниже скрипт.

```
#!/usr/bin/python
f1=int(input("Enter number of the first frame : "))
f2=int(input("Enter number of the last frame : "))
n1=input("Enter name of new file : ")
f = open(n1, "w")
f.write("[ Cluster_frames ]\n")
for i in range(f1+1, f2+2):
  f.write("%i\n"%(i))
f.close()
```

Далее выполняется команда GROMACS:

```
echo 0 | gmx trjconv -s cluster1.pdb -f ordered.xtc -fr cluster1.ndx -o cluster1.xtc
```

3) Для расчета попарных расстояний между всеми аминокислотных остатков двух белков можно применить команду GROMACS:

```
gmx pairdist -s cluster1.pdb -f cluster1.xtc -n chains.ndx -o dist1.xvg -refgrouping res -selgrouping res -ref chB -sel chA
```

4) Для анализа полученной матрицы потребуется последовательность аминокислотных остатков, входящих в каждый из исследуемых белков. Получить ее можно с применением скрипта, приведенного ниже:

```
#!/usr/bin/python
import sys
if len(sys.argv) <= 1:
   print('usage: python pdb2fasta.py file.pdb > file.fasta')
   exit()
input_file = open(sys.argv[1])
letters = {'ALA':'A', 'ARG':'R', 'ASN':'N', 'ASP':'D', 'CYS':'C', 'GLU':'E', 'GLN':'Q', 'GLY':'G',
'HIS':'H', 'HSD':'H', 'HSE':'H', 'ILE':'I', 'LEU':'L', 'LYS':'K', 'MET':'M', 'PHE':'F', 'PRO':'P',
'SER':'S', 'THR':'T','TRP':'W', 'TYR':'Y', 'VAL':'V'}
print('>',sys.argv[1])
prev2 = 'X'
prev3 = '-1'
for line in input_file:
   toks = line.split()
   if len(toks)<1: continue
   if toks[0] != 'ATOM': continue
   if toks[4] != prev2:
      sys.stdout.write('\n>chain_%c\n'%toks[4])
   if toks[5] != prev3:
      sys.stdout.write('%c' % letters[toks[3]])
   prev2 = toks[4]
   prev3 = toks[5]
sys.stdout.write('\n')
input_file.close()
```

5) Для получения ранжированного списка контактов в кластере можно применить скрипт, приведенный ниже:

```
#!/usr/bin/python
import numpy as np
distances = np.loadtxt('cluster2_dist.xvg', skiprows=24)
sequence1 =
'VEVLLGGGDGSLAFLPGDFSVASGEEIVFKNNAGFPHNVVFDEDEIPSGVDAAKISMSEED
LLNAPGETYKVTLTEKGTYKFYCSPHQGAGMVGKVTVN'
sequence2 =
'YPIFAQQNYENPREATGRIVCANCHLASKPVDIEVPQAVLPDTVFEAVVKIPYDMQLKQVLA
NGKKGALNVGAVLILPEGFELAPPDRISPEMKEKIGNLSFQNYRPNKKNILVIGPVPGQKYSE
ITFPILAPDPATNKDVHFLKYPIYVGGNRGRGQIYPDGSKSNNTVYNATAGGIISKILRKEKG
GYEITIVDASNERQVIDIIPRGLELLVSEGESIKLDQPLTSNPNVGGFGQGDAEIVLQDPLR'
distances3d = distances[:,1:].reshape((distances.shape[0], len(sequence1), -1))
s1=np.sum(distances3d<0.5,axis=0)
sort1=s1[s1.nonzero()].argsort()[::-1]
con1=list(zip(np.argwhere(s1)[sort1],s1[s1.nonzero()][sort1]/float(distances3d.shape[0])))
for i in con1:
   print('%s%d -- %s%d\t%4.2f'%(sequence1[i[0][0]],i[0][0]+1,sequence2[i[0][1]],i[0][1]+1, i[1]
))
```

6) Для построения графика изменения числа контактов можно применить скрипт:

```
#!/usr/bin/python
```

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import host_subplot
from mpl_toolkits import axisartist
distances = np.loadtxt('dist1.xvg', skiprows=23)
distances_fe = np.loadtxt('cu_fe_distance.xvg', skiprows=23)
sequence1 =
'VEVLLGGGDGSLAFLPGDFSVASGEEIVFKNNAGFPHNVVFDEDEIPSGVDAAKISMSEED
LLNAPGETYKVTLTEKGTYKFYCSPHQGAGMVGKVTVNX'
sequence2 =
'YPIFAQQNYENPREATGRIVCANCHLASKPVDIEVPQAVLPDTVFEAVVKIPYDMQLKQVLA
NGKKGALNVGAVLILPEGFELAPPDRISPEMKEKIGNLSFQNYRPNKKNILVIGPVPGQKYSE
ITFPILAPDPATNKDVHFLKYPIYVGGNRGRGQIYPDGSKSNNTVYNATAGGIISKILRKEKG
GYEITIVDASNERQVIDIIPRGLELLVSEGESIKLDQPLTSNPNVGGFGQGDAEIVLQDPLRX'
distances3d = distances[:,1:].reshape((distances.shape[0], len(sequence1), -1))
contacts3d = distances3d < 0.5
hydrophobic = 'GAVLIFMCYW'
def mask_chars(st, chs):
    return np.argwhere(np.isin(np.frombuffer(st.encode(), dtype=np.uint8),
np.frombuffer(chs.encode(), dtype=np.uint8))).ravel()
def get_masked(c3d, chs1, chs2, sequence1=sequence1, sequence2=sequence2):
    return c3d.take(mask_chars(sequence1, chs1), axis=1).take(mask_chars(sequence2,
chs2), axis=2)
def moving_average(x, w):
    return np.convolve(x, np.ones(w), 'valid') / w
plt.figure(figsize=[10,5])
host = host_subplot(111, axes_class=axisartist.Axes)
plt.subplots_adjust(right=0.70)

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()

par2.axis["right"] = par2.new_fixed_axis(loc="right", offset=(50, 0))
par3.axis["right"] = par3.new_fixed_axis(loc="right", offset=(100, 0))

par1.axis["right"].toggle(all=True)
par2.axis["right"].toggle(all=True)
par3.axis["right"].toggle(all=True)

p1, = host.plot(moving_average(distances_fe[:,0]/1000, 20),
moving_average(distances_fe[:,1], 20), label="Cu-Fe distance", linewidth=3)
p2, = par1.plot(moving_average(distances_fe[:,0]/1000, 20),
moving_average(contacts3d.sum(axis=(1,2)),20), label="All contacts")
p3, = par2.plot(moving_average(distances_fe[:,0]/1000, 20),
moving_average(get_masked(contacts3d, 'DE', 'KR').sum(axis=(1,2)) +
        get_masked(contacts3d, 'KR', 'DE').sum(axis=(1,2)), 20), label="Electrostatic
contacts")
```

```
p4, = par3.plot(moving_average(distances_fe[:,0]/1000,
20),moving_average(get_masked(contacts3d, hydrophobic, hydrophobic).sum(axis=(1,2)),
20), label="Hydrophobic contacts")
host.set_xlim(0, max(distances_fe[:,0])/1000)
host.set_ylim(min(distances_fe[:,1]), max(distances_fe[:,1]))
par1.set_ylim(-0.1, max(contacts3d.sum(axis=(1,2))))
par2.set_ylim(-0.1, max(get_masked(contacts3d, 'DE', 'KR').sum(axis=(1,2)) +
      get_masked(contacts3d, 'KR', 'DE').sum(axis=(1,2))))
par3.set_ylim(-0.1, max(get_masked(contacts3d, hydrophobic,
hydrophobic).sum(axis=(1,2))))

host.set_xlabel("Time, ns")
host.set_ylabel("Cu-Fe distance")
par1.set_ylabel("All contacts")
par2.set_ylabel("Electrostatic contacts")
par3.set_ylabel("Hydrophobic contacts")

host.legend()

host.axis["left"].label.set_color(p1.get_color())
par1.axis["right"].label.set_color(p2.get_color())
par2.axis["right"].label.set_color(p3.get_color())
par3.axis["right"].label.set_color(p4.get_color())

plt.show()
```

7) Для расчета количества воды вблизи гидрофобных аминокислотных остатков двух белков можно применить команду GROMACS:

```
gmx select -s topol.tpr -f traj.xtc -os water_around_system_5A.xvg -n
hydrophobic_amino.ndx -select "name OW and within 0.5 of group Hydrophobic_System"
gmx select -s topol.tpr -f traj.xtc -os water_around_chA_5A.xvg -n hydrophobic_amino.ndx -
select "name OW and within 0.5 of group Hydrophobic_chA"
gmx select -s topol.tpr -f traj.xtc -os water_around_chB_5A.xvg -n hydrophobic_amino.ndx -
select "name OW and within 0.5 of group Hydrophobic_chB"
```

8) Для построения графика изменения числа молекул воды можно применить скрипт:

```
#!/usr/bin/python
import pandas as pd
from mpl_toolkits.axes_grid1 import host_subplot
from mpl_toolkits import axisartist
import matplotlib.pyplot as plt
base1= 'cu_fe_distance'
distances_fe = pd.read_csv(base1+'.xvg', sep='\s+', skiprows=24,header=None)
base2= 'water_around'
water_molA_hyd = pd.read_csv(base2+'_chA_5A.xvg', sep='\s+',
skiprows=23,header=None)
```

```
water_molB_hyd = pd.read_csv(base2+'_chB_5A.xvg', sep='\s+',
skiprows=23,header=None)
water_molS_hyd = pd.read_csv(base2+'_system_5A.xvg', sep='\s+',
skiprows=23,header=None)
def moving_average(x, w):
    return np.convolve(x, np.ones(w), 'valid') / w
plt.figure('Mix2',figsize=[10,5])
host = host_subplot(111, axes_class=axisartist.Axes)
plt.subplots_adjust(right=0.70)

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()
par4 = host.twinx()

par2.axis["right"] = par2.new_fixed_axis(loc="right", offset=(50, 0))
par3.axis["right"] = par3.new_fixed_axis(loc="right", offset=(100, 0))
par4.axis["right"] = par4.new_fixed_axis(loc="right", offset=(150, 0))

par1.axis["right"].toggle(all=True)
par2.axis["right"].toggle(all=True)
par3.axis["right"].toggle(all=True)
par4.axis["right"].toggle(all=True)
n_len=10
p1, = host.plot(moving_average(distances_fe[0]/1000, n_len),
moving_average(distances_fe[1], n_len), label="Fe-Fe distance", linewidth=3)
p2, =
par1.plot(moving_average(water_molS_hyd[0]/1000,n_len),moving_average(water_molS_hy
d[1], n_len), label="Water around all hydrophobic amino acids")
p3, =
par2.plot(moving_average(water_molA_hyd[0]/1000,n_len),moving_average(water_molA_hy
d[1], n_len), label="Water around Pc hydrophobic amino acids")
p4, =
par3.plot(moving_average(water_molB_hyd[0]/1000,n_len),moving_average(water_molB_hy
d[1], n_len), label="Water around CytF hydrophobic amino acids")
p5, =
par4.plot(moving_average(water_molS_hyd[0]/1000,xz1),moving_average(water_molA_hyd[
1]+water_molB_hyd[1]-water_molS_hyd[1], n_len), label="Water around (CytF+Pc) - all
hydrophobic amino acids")
host.set_xlim(0, 1000)
host.set_ylim(1, 4.5)
par1.set_ylim(1000, 1170)
par2.set_ylim(300, 375)
par3.set_ylim(700, 850)
par4.set_ylim(-5, 50)
host.set_xlabel("Time, ns")
host.set_ylabel("Cu-Fe distance")
par1.set_ylabel("Water around all hydrophobic amino acids")
```

```
par2.set_ylabel("Water around Pc hydrophobic amino acids")
par3.set_ylabel("Water around CytF hydrophobic amino acids")
par4.set_ylabel("Water around (CytF+Pc) - all hydrophobic amino acids")
host.legend()
host.axis["left"].label.set_color(p1.get_color())
par1.axis["right"].label.set_color(p2.get_color())
par2.axis["right"].label.set_color(p3.get_color())
par3.axis["right"].label.set_color(p4.get_color())
par4.axis["right"].label.set_color(p5.get_color())
plt.show()
```